Logistic Regression

전 재 욱

Embedded System 연구실
성균관대학교

Outline

- Classification
- Hypothesis representation
- Decision boundary
- Cost function
- Derivation of cost function via MLE
- Simplified cost function and gradient descent
- Advanced optimization
- Multi-class classification

Outline

- Classification
- Hypothesis representation
- Decision boundary
- Cost function
- Derivation of cost function via MLE
- Simplified cost function and gradient descent
- Advanced optimization
- Multi-class classification

Classification

- Email
  - Spam / Not Spam?

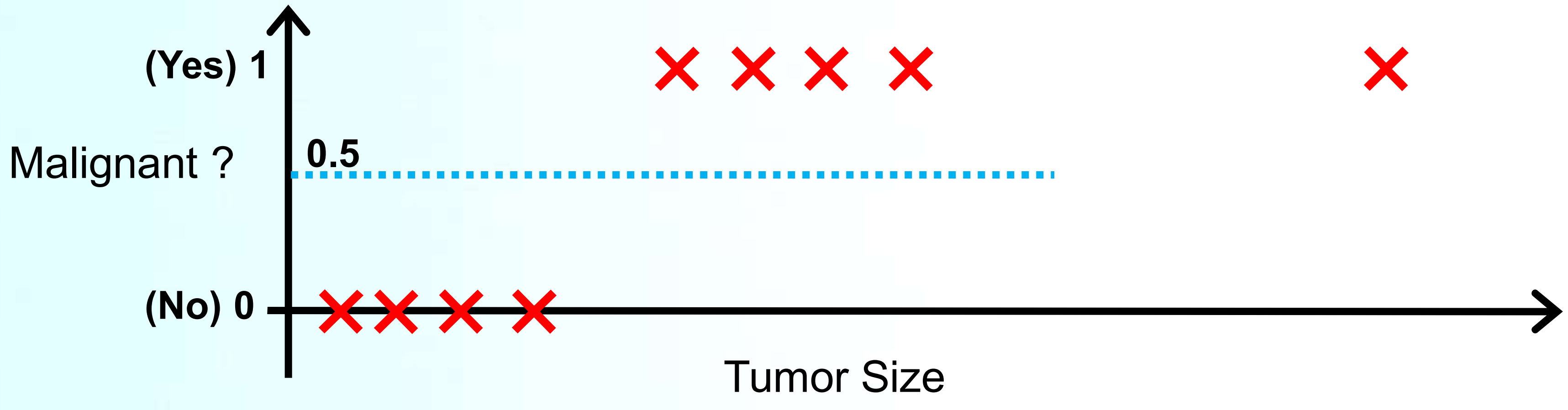
- Online Transactions
  - Fraudulent (Yes / No)?

- Tumor
  - Malignant / Benign ?

- $y \in \{0, 1\}$
  - 0: “Negative Class” (e.g., benign tumor)
  - 1: “Positive Class” (e.g., malignant tumor)

- $y \in \{0, 1, 2, 3\}$

Classification



- Threshold classifier output $h_\theta(x)$ at 0.5:
  - If $h_\theta(x) \geq 0.5$, predict "$y = 1$"
  - If $h_\theta(x) < 0.5$, predict "$y = 0$"

Classification vs Logistic Regression

- Classification: $y = 0$ or 1
  - If we were to use a linear regression for classification, $h_\theta(x)$ can be > 1 or < 0
- Logistic Regression
  - Always $0 \leq h_\theta(x) \leq 1$
  - One classification algorithm

Outline

- Classification
- Hypothesis representation
- Decision boundary
- Cost function
- Derivation of cost function via MLE
- Simplified cost function and gradient descent
- Advanced optimization
- Multi-class classification

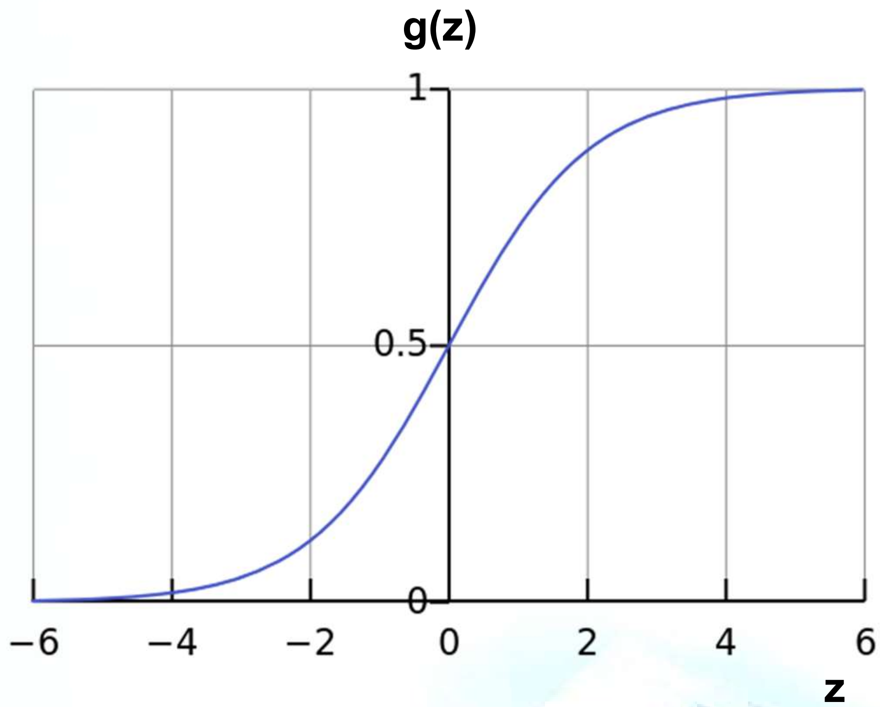
Logistic Regression Model

- Takes a probabilistic approach
  to learning discriminative functions (i.e., a classifier)

  - $h_\theta(x)$ should give $p(y=1|x;\theta)$
    - Want $0 \le h_\theta(x) \le 1$

Logistic Regression Model

- Want $0 \leq h_\theta(x) \leq 1$
  - $h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{(-\theta^T x)}}$
- $g(z) = \frac{1}{1+\exp(-z)}$
  - Sigmoid function
  - Logistic function



Interpretation of Hypothesis Output

- $h_\theta(x)$
  - *Estimated probability that $y = 1$ on input x*
    - Example
      - If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ tumor\_size \end{bmatrix}$ and $h_\theta(x) = 0.7$

        tell patient that 70% chance of tumor being malignant

- $h_\theta(x) = P(y = 1|x;\theta)$
  - "probability that $y = 1$, given x, parameterized by θ"

- $y = 0$ or 1
  - $P(y = 0|x;\theta) + P(y = 1|x;\theta) = 1$
    - $P(y = 0|x;\theta) = 1 - P(y = 1|x;\theta)$

Another Interpretation

- $h_\theta(x)$
  - <u>*Estimated probability that $y = 1$ on input x*</u>

Odds of y =1

$$\log \boxed{\frac{p(y=1|x;\theta)}{p(y=0|x;\theta)}} = \log \frac{h}{1-h} = \log \frac{\frac{1}{1+\exp(-\theta^T x)}}{\frac{\exp(-\theta^T x)}{1+\exp(-\theta^T x)}}$$

$$= \log \frac{1}{\exp(-\theta^T x)} = \theta^T x \; = \; \theta_0 + \theta_1 x_1 + \cdots + \; \theta_n x_n$$

- Logistic regression assumes that
  - the {log odds} is a linear function of x

Embedded System Lab.

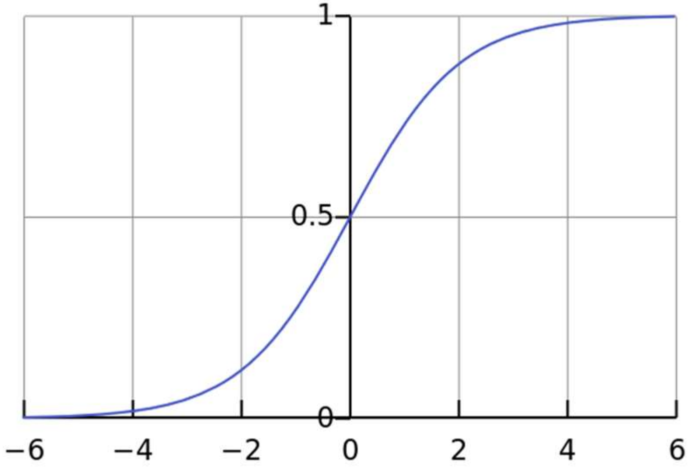
Outline

- Classification
- Hypothesis representation
- Decision boundary
- Cost function
- Derivation of cost function via MLE
- Simplified cost function and gradient descent
- Advanced optimization
- Multi-class classification

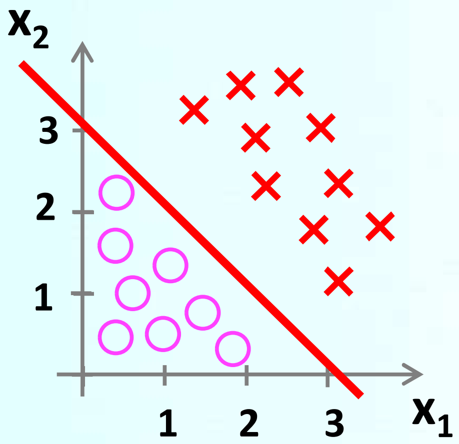
Logistic Regression

- $h_\theta(x) = g(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$

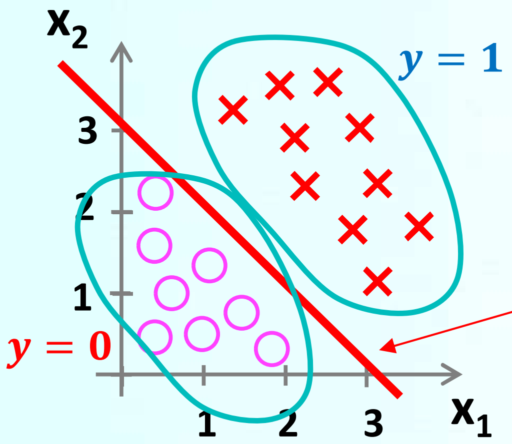
- Suppose
  - predict "$y = 1$" if $h_\theta(x) \geq 0.5$
    - $\theta^T x \geq 0$
  - predict "$y = 0$" if $h_\theta(x) < 0.5$
    - $\theta^T x < 0$



Decision Boundary



Decision Boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

- $\theta_0 = -3,\ \theta_1 = 1,\ \theta_2 = 1$

- Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$
  - i.e. if $x_1 + x_2 \geq 3$
    - $h_\theta(x) = 0.5$ when $x_1 + x_2 = 3$

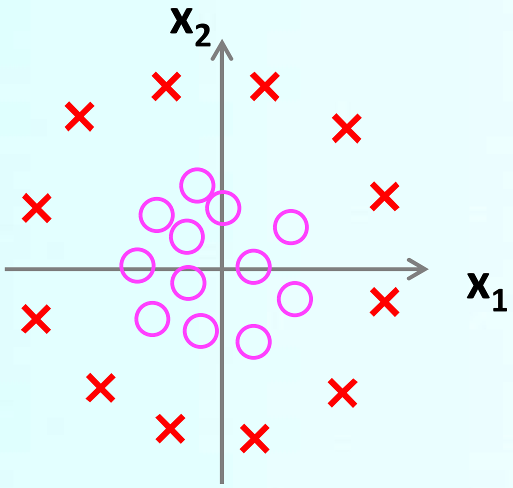
- Predict "$y = 0$" if $x_1 + x_2 < 3$

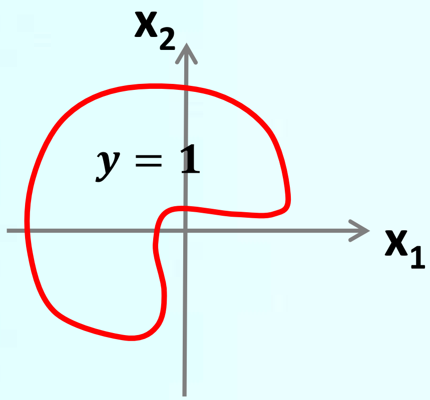
Nonlinear Decision Boundary

- Can apply basis function expansion to features,
  same as with linear regression

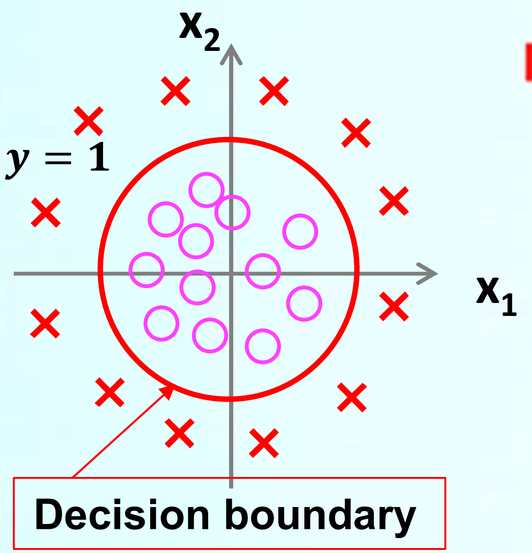
$$x = \begin{bmatrix} 1 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ \dots \\ x_n \end{bmatrix}$$

Nonlinear Decision Boundary

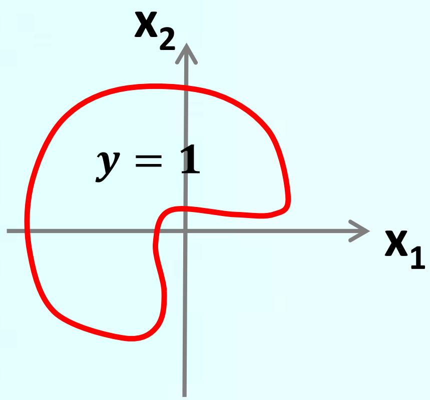




Nonlinear Decision Boundary



- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$
  - Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$
    - $\theta_0 = -1,\ \theta_1 = \theta_2 = 0, \theta_3 = \theta_4 = 1$
    - $\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$



- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \cdots)$

Outline

- Classification
- Hypothesis representation
- Decision boundary
- Cost function
- Derivation of cost function via MLE
- Simplified cost function and gradient descent
- Advanced optimization
- Multi-class classification

Parameter θ

- Training set
  - m examples: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$
  - For each example, $x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \cdots \\ x_n^{(i)} \end{bmatrix} \in R^{n+1}$, $x_0^{(i)} = 1$,

    $y \in \{0, 1\}$

- $h_\theta(x) = g(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$
  - How to choose parameters θ ?

Cost Function

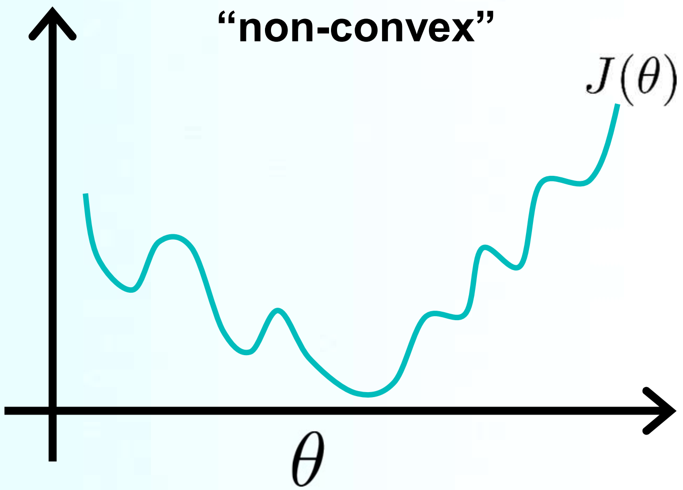
- Linear regression
  - $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

- Define
  - $Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) = \frac{1}{2}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$
  - $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$

- Then
  - $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(\frac{1}{1+\exp(-\theta^T x^{(i)})} - y^{(i)}\right)^2$

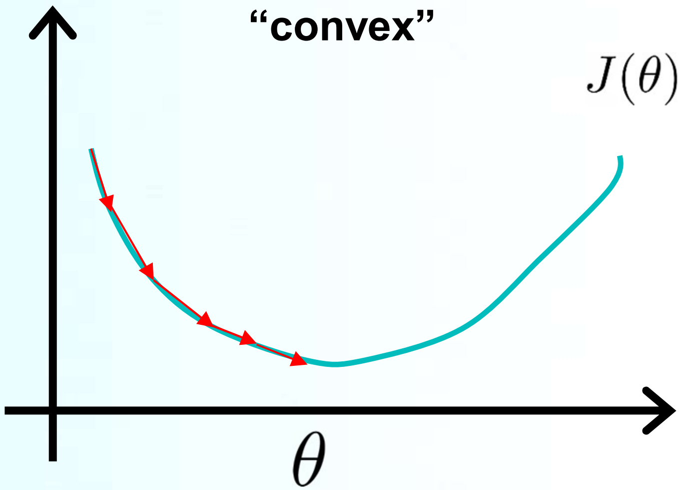
Cost Function

- Then
  - $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(\frac{1}{1+\mathrm{ex}\ \ (-\theta^T x^{(i)})} - y^{(i)}\right)^2$
    - Non-convex function
      - ➢ Many local min
      - ➔ Gradient descent may not find the global min



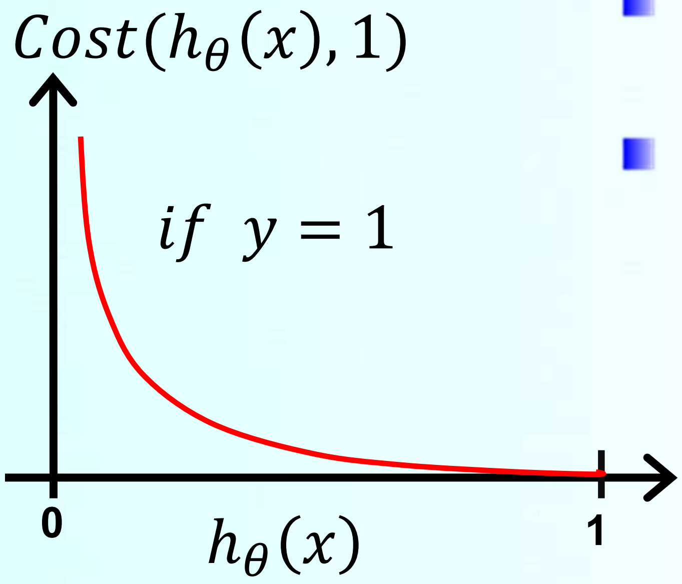
Cost Function

- If $J(\theta)$ is convex
  - Gradient descent can converge the global minimum



Convex Logistic Regression Cost Function

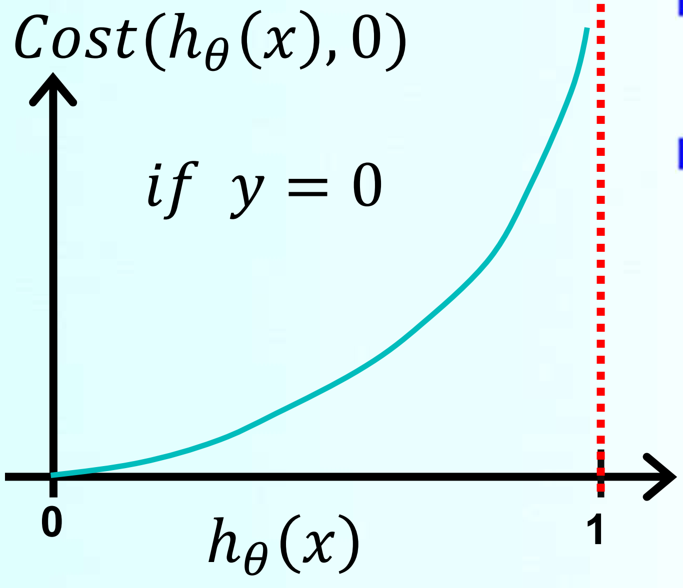
- Convex logistic regression
  - $Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & if\ \ y = 1 \\ -\log(1 - h_\theta(x)) & if\ \ y = 0 \end{cases}$



- $Cost = 0$ if $y = 1$ and $h_\theta(x) = P(y = 1|x;\ \theta)$=1
- $Cost \rightarrow \infty$ as $h_\theta(x) \rightarrow 0$
  - Captures intuition that
    - if $h_\theta(x) = 0$ (predict $P(y = 1|x; \theta) = 0$) but $y = 1$ this learning algorithm will be penalized by a very large cost

Convex Logistic Regression Cost Function

- $Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & if \;\; y = 1 \\ -\log(1 - h_\theta(x)) & if \;\; y = 0 \end{cases}$



- $Cost = 0$ if $y = 0$ and $h_\theta(x) = P(y = 1|x;\ \theta)=0$
- $Cost \to \infty$ as $h_\theta(x) \to 1$
  - Captures intuition that
    - if $h_\theta(x) = 1$ (predict $P(y = 1|x;\theta) = 1$) but $y = 0$ this learning algorithm will be penalized by a very large cost

Convex Logistic Regression Cost Function

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & if \;\; y = 1 \\ -\log(1 - h_\theta(x)) & if \;\; y = 0 \end{cases}$$

- Always $y = 0$ or 1
  - So, for each training example $(x^{(i)}, y^{(i)})$,
    - $Cost(h_\theta(x^{(i)}), y^{(i)}) = -y^{(i)} \log(h_\theta(x)) - (1 - y^{(i)}) \log(1 - h_\theta(x))$
  - For all training examples $(x^{(i)}, y^{(i)})$ for $i = 1, 2, \ldots, m$
    - $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$

$$= \; -\frac{1}{m} \left[\sum_{i=1}^{m} \{ \; y^{(i)} \log h_\theta \left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log(1 - h_\theta \left(x^{(i)}\right)) \; \} \right]$$

Convex Logistic Regression Cost Function

- Logistic regression cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m}\{\ y^{(i)}\log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log(1 - h_\theta\left(x^{(i)}\right))\ \}\right]$$

- To fit parameters θ
  - $min_\theta J(\theta)$

- To make a prediction, given new x_{new}:
  - $h_\theta(x_{new}) = \frac{1}{1+\exp(-\theta^T x_{new})} = P(y = 1|\ x_{new}; \theta)$

Outline

- Classification
- Hypothesis representation
- Decision boundary
- Cost function
- Derivation of cost function via MLE
- Simplified cost function and gradient descent
- Advanced optimization
- Multi-class classification

Derivation of Cost Function via MLE

- $h_\theta(x)$
  - *Estimated probability that $y = 1$ on input x*
    - $p(y = 1|x; \theta) = h_\theta(x)$, $p(y = 0|x; \theta) = 1 - h_\theta(x)$
      - $p(y|x; \theta) = \left(h_\theta(x)\right)^y \left(1 - h_\theta(x)\right)^{1-y}$

- Given m (independent) training examples,
  the likelihood of the parameters, $L(\theta)$,

$$L(\theta) = \prod_{i=1}^{m} p\left(y^{(i)}|x^{(i)}; \theta\right) = \prod_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right)\right)^{y^{(i)}} \left(1 - h_\theta\left(x^{(i)}\right)\right)^{1-y^{(i)}}$$

Derivation of Cost Function via MLE

- θ_{MLE}
  - which maximizes the likelihood (Maximum likelihood estimation)

$$\theta_{MLE} = arg\max_{\theta} L(\theta) = arg\max_{\theta} \prod_{i=1}^{m} p\left(y^{(i)} \middle| x^{(i)}; \theta\right)$$

$$= arg\max_{\theta} \prod_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right)\right)^{y^{(i)}} \left(1 - h_\theta\left(x^{(i)}\right)\right)^{1-y^{(i)}}$$

Derivation of Cost Function via MLE

- θ_{MLE}
  - which maximizes the likelihood (Maximum likelihood estimation)

Or

$$\theta_{MLE} = arg\max_{\theta} \log L(\theta) = arg\max_{\theta} \log \prod_{i=1}^{m} p\left(y^{(i)} \middle| x^{(i)}; \theta\right)$$

$$= arg\max_{\theta} \sum_{i=1}^{m} \log p\left(y^{(i)} \middle| x^{(i)}; \theta\right)$$

$$= arg\max_{\theta} \sum_{i=1}^{m} \log \left(h_{\theta}\left(x^{(i)}\right)\right)^{y^{(i)}} \left(1 - h_{\theta}\left(x^{(i)}\right)\right)^{1-y^{(i)}}$$

$$= arg\max_{\theta} \sum_{i=1}^{m} \left\{ y^{(i)} \log h_{\theta}\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_{\theta}\left(x^{(i)}\right)\right) \right\}$$

Derivation of Cost Function via MLE

- θ_{MLE}
  - which maximizes the likelihood (Maximum likelihood estimation)

$$\theta_{MLE} = \underset{\theta}{argmax} \sum_{i=1}^{m} \{ \ y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right) \ \}$$

➔ Logistic regression cost function $J(\theta)$

$$J(\theta) = \ - \ \frac{1}{m} \sum_{i=1}^{m} \{ \ y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right) \ \}$$

➔ To fit parameters θ, find θ_{MLE} s.t. $\theta_{MLE} = \min_{\theta} J(\theta)$

Outline

- Classification
- Hypothesis representation
- Decision boundary
- Cost function
- Derivation of cost function via MLE
- Simplified cost function and gradient descent
- Advanced optimization
- Multi-class classification

Logistic Regression Cost Function

- Logistic regression cost function

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\{\ y^{(i)}\log h_\theta\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log(1-h_\theta\left(x^{(i)}\right))\ \}\right]$$

- To fit parameters θ
  - $min_\theta J(\theta)$

- To make a prediction, given newx_{new}:
  - $h_\theta(x_{new}) = \frac{1}{1+\exp(-\theta^T x_{new})} = P(y=1|x_{new};\theta)$

Gradient Descent

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\{\; y^{(i)}\log h_\theta\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log(1-h_\theta\left(x^{(i)}\right))\;\}\right]$$

- Want $min_\theta J(\theta)$
  - Gradient descent

  Repeat {

$$\theta_j \leftarrow \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

  } (simultaneously update for every θ_j)

  - $\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$

Gradient Descent

- $\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j}\left(- \ \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log(1 - h_\theta\,(x^{(i)}))\right]\right)$

- $\frac{\partial}{\partial \theta_j} h_\theta\left(x^{(i)}\right)$

Gradient Descent

- $\frac{\partial}{\partial\theta_j}J(\theta) = \frac{\partial}{\partial\theta_j}\left(- \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log(1 - h_\theta(x^{(i)}))\right]\right)$

- $\frac{\partial}{\partial\theta_j}\log h_\theta\left(x^{(i)}\right)$

Gradient Descent

- $\frac{\partial}{\partial\theta_j}J(\theta) = \frac{\partial}{\partial\theta_j}\left(-\ \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log(1-h_\theta\,(x^{(i)}))\right]\right)$

- $\frac{\partial}{\partial\theta_j}\log\left(1-h_\theta(x^{(i)})\right)$

Gradient Descent

- $\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j}\left(- \ \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log(1 - h_\theta\,(x^{(i)}))\right]\right)$

- $\frac{\partial}{\partial \theta_j} J(\theta) =$

Gradient Descent

- $\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j}\left(- \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log(1 - h_\theta\,(x^{(i)}))\right]\right)$

- $\frac{\partial}{\partial \theta_j} h = \frac{1}{h}\left(x_j^{(i)} h(1-h)\right)$
- $\frac{\partial}{\partial \theta_j} \log h_\theta\left(x^{(i)}\right) = \frac{1}{h}\frac{\partial}{\partial \theta_j} h = \frac{1}{h}\left(x_j^{(i)} h(1-h)\right)$
- $\frac{\partial}{\partial \theta_j} \log\left(1 - h_\theta(x^{(i)})\right) = \frac{1}{1-h}\frac{\partial}{\partial \theta_j}(1-h) = \frac{1}{1-h}\left(x_j^{(i)} h(1-h)\right)$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} \frac{1}{h} x_j^{(i)} h(h-1) + \left(1 - y^{(i)}\right)\frac{1}{1-h}\left(x_j^{(i)} h(h-1)\right)\right)$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} x_j^{(i)}(h-1) - \left(1 - y^{(i)}\right) x_j^{(i)} h\right)$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)} x_j^{(i)} + x_j^{(i)} h\right) = \frac{1}{m}\sum_{i=1}^{m}\left((h - y^{(i)}) x_j^{(i)}\right)$$

Gradient Descent

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\{\ y^{(i)}\log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log(1 - h_\theta\left(x^{(i)}\right))\ \}\right]$$

- Want $min_\theta J(\theta)$
  - Gradient descent

  Repeat {

$$\theta_j \leftarrow \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(\ h_\theta\left(x^{(i)}\right) - \ y^{(i)}\ \right)x_j^{(i)}$$

  } (simultaneously update for every θ_j)

  - Algorithm looks identical to linear regression!
    - Logistic regression: $h_\theta(z) = \frac{1}{1+exp(-\theta^T x)}$
    - Linear regression: $h_\theta(z) = \theta^T x$

Outline

- Classification
- Hypothesis representation
- Decision boundary
- Cost function
- Derivation of cost function via MLE
- Simplified cost function and gradient descent
- Advanced optimization
- Multi-class classification

Optimization Algorithm

- Cost function $J(\theta)$

- Want $min_{\theta} J(\theta)$

- Given θ, we have code that can compute
  - $J(\theta)$
  - $\frac{\partial}{\partial\theta} J(\theta)$ (for $j = 0, 1, \ldots, n$)

- Gradient descent

  Repeat {

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial\theta_j} J(\theta)$$

  } (simultaneously update for every θ_j)

Optimization Algorithm

- Given θ, we have code that can compute
  - $J(\theta)$
  - $\frac{\partial}{\partial\theta}J(\theta)$ (for $j = 0, 1, \dots, n$)

- Optimization algorithms
  - Gradient descent
  - Conjugate gradient
  - BFGS
    - Broyden-Fletcher-Goldfarb-Shanno
  - L-BFGS
    - Limited memory - BFGS

- Advantages
  - No need to manually pick α
  - Often faster than gradient descent
- Disadvantages:
  - More complex

Outline

- Classification
- Hypothesis representation
- Decision boundary
- Cost function
- Derivation of cost function via MLE
- Simplified cost function and gradient descent
- Advanced optimization
- Multi-class classification

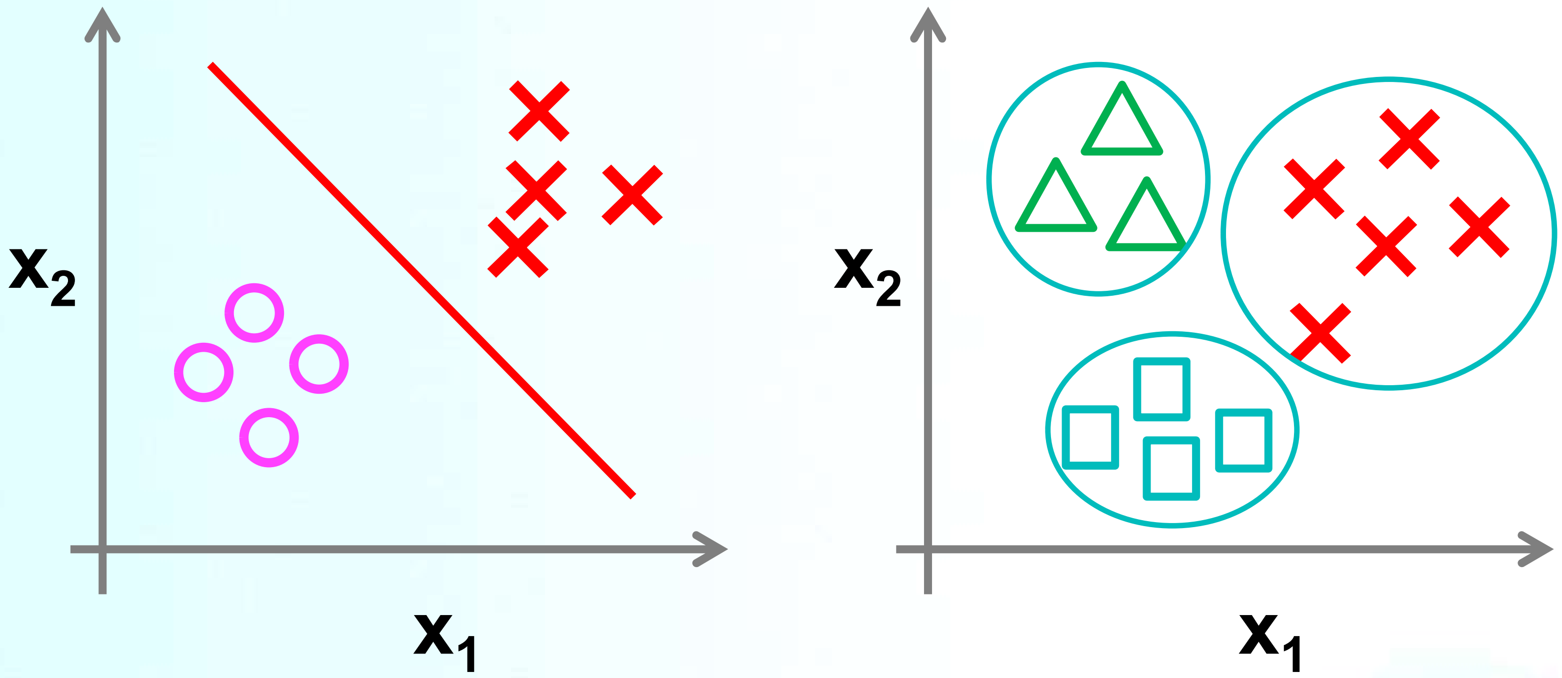
Multiclass Classification

- Email foldering/tagging
  - Work($y = 1$), Friends ($y = 2$), Family ($y = 3$), Hobby ($y = 4$)

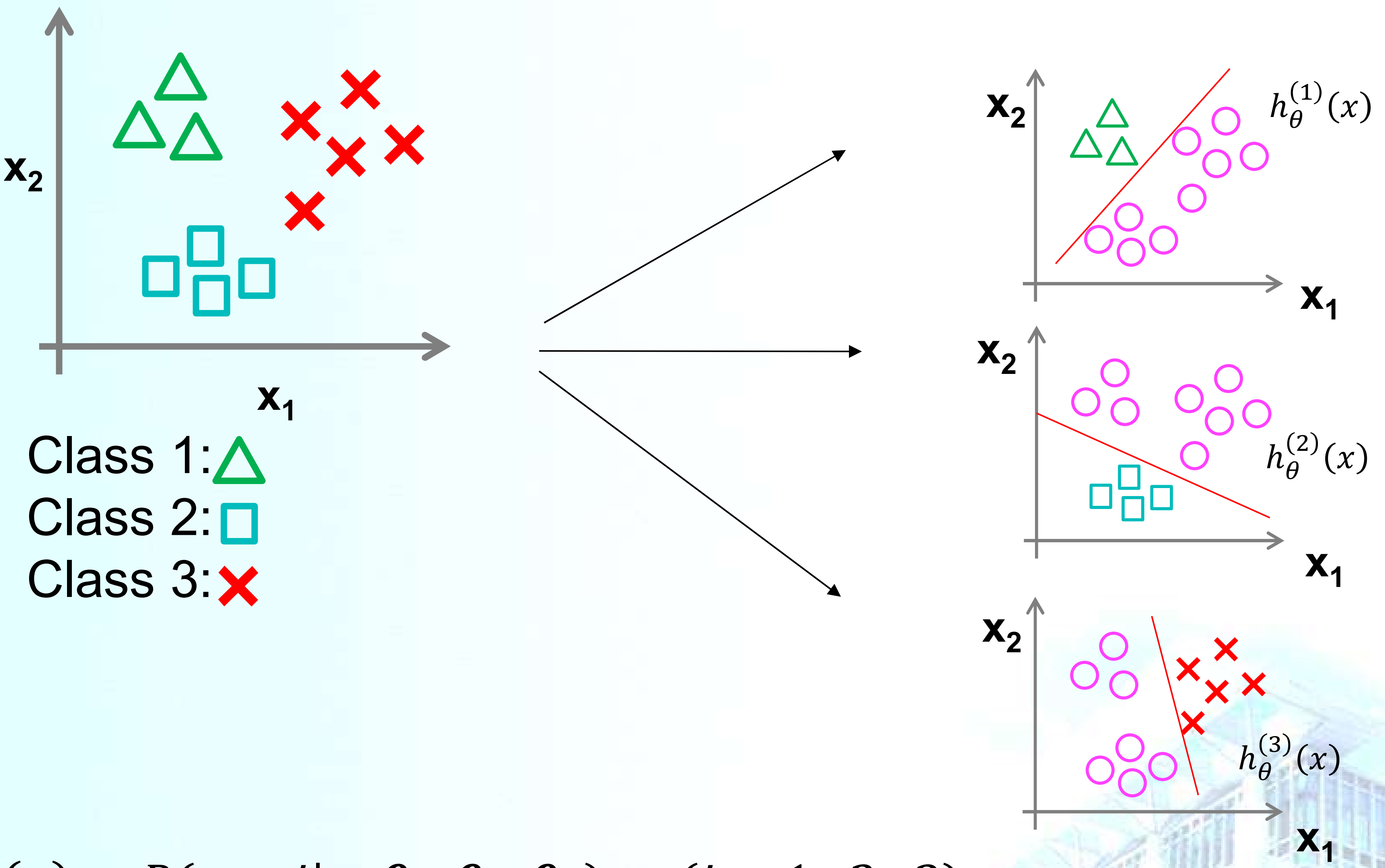
- Medical diagrams
  - Not ill ($y = 1$), Cold ($y = 2$), Flu ($y = 3$),

- Weather
  - Sunny ($y = 1$), Cloudy ($y = 2$), Rain ($y = 3$), Snow ($y = 4$),

Binary vs Multiclass Classification



One vs All (One vs Rest)



$$h_{\boldsymbol{\theta}_i}^{(i)}(\boldsymbol{x}) = P(y = i | \boldsymbol{x}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) \qquad (i = 1,\ 2,\ 3)$$

One vs All

- Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$.

- On a new input x_{new}, to make a prediction, pick the class i that maximizes

$$max_i h_\theta^{(i)}(x_{new})$$

Multi-Class Logistic Regression

- $h_\theta(x)$
  - <u>*Estimated probability that $y = 1$ on input x*</u>
  - $p(y = 1|x; \theta)$

- For binary classes

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)} = \frac{\exp(\theta^T x)}{\boxed{1} + \boxed{\exp(\theta^T x)}}$$

Weight assigned to y=0 **Weight assigned to y=1**

Multi-Class Logistic Regression

- For *K* classes $\{1,2,\cdots,K\}$:

$$p(y=j|\boldsymbol{x};\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_K) = \frac{\exp(\boldsymbol{\theta}_j^T\boldsymbol{x})}{\sum_{k=1}^{K}\exp(\boldsymbol{\theta}_k^T\boldsymbol{x})}$$

  - Called the softmax function

- Given a sample vector $\boldsymbol{x}$
  - Estimated probability for the $j'th$ class

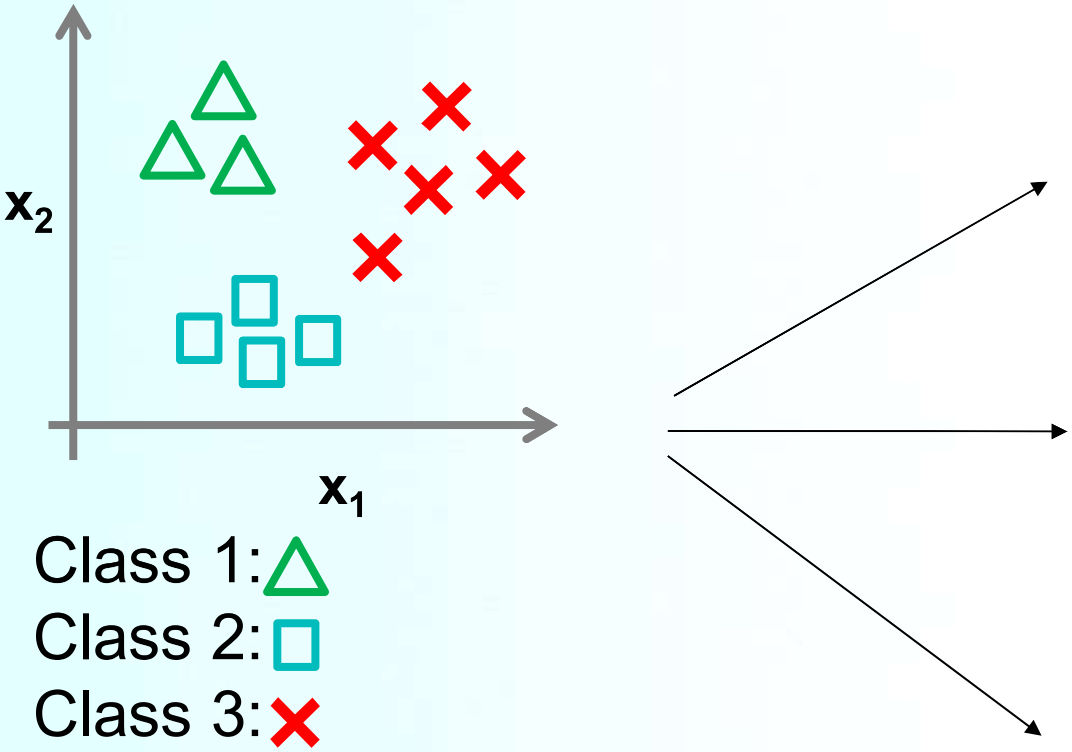
$$h_\theta^{(j)}(\boldsymbol{x}) = p(y=j|\boldsymbol{x};\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_K) = \frac{\exp(\boldsymbol{\theta}_j^T\boldsymbol{x})}{\sum_{k=1}^{K}\exp(\boldsymbol{\theta}_k^T\boldsymbol{x})} \quad (j=1,\cdots,K)$$

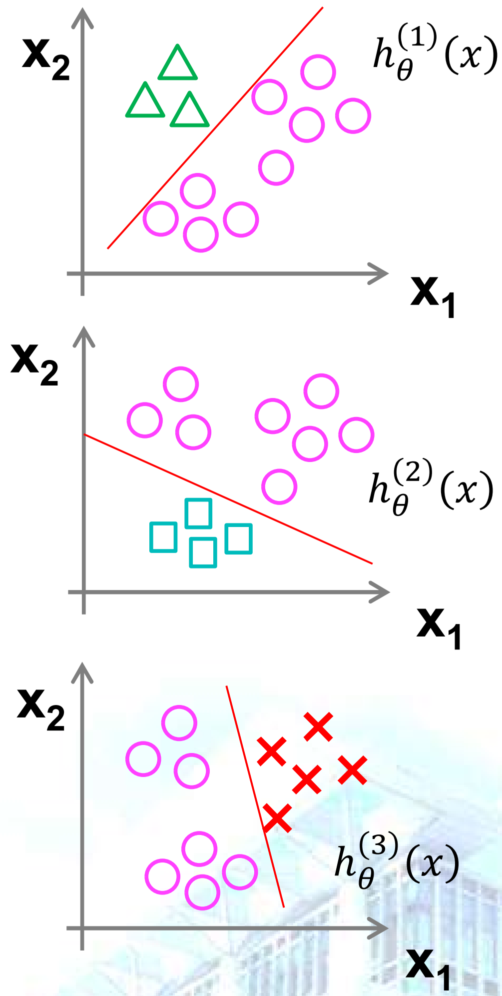
Softmax Function (Normalized Exponential Function)

- $p(y=j|\boldsymbol{x};\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_K) = \dfrac{\exp(\boldsymbol{\theta}_j^T\boldsymbol{x})}{\sum_{k=1}^{K}\exp(\boldsymbol{\theta}_k^T\boldsymbol{x})}$
  - A generalization of the logistic regression that
    maps n-dim vector x of real values
    to K-dim vector p of real values in the range (0,1)
  - $\sum_{j=1}^{K} p(y=j|\boldsymbol{x};\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_K) = 1$
  - The output of softmax function can be used to
    - represent a categorical distribution
      - Probability distribution over K different possible outcomes

- $$h_\theta(x) = \begin{bmatrix} h_\theta^{(1)} \\ h_\theta^{(2)} \\ \cdots \\ h_\theta^{(K)} \end{bmatrix} = \begin{bmatrix} p(y=1|\boldsymbol{x};\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_K) \\ p(y=2|\boldsymbol{x};\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_K) \\ \cdots \\ p(y=K|\boldsymbol{x};\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_K) \end{bmatrix}$$

Multi-Class Logistic Regression





- Train a logistic regression classifier for each class j to predict the probability that y = j with

$$h_\theta^{(j)}(\boldsymbol{x}) = p(y = j|\boldsymbol{x}; \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_K) = \frac{\exp(\boldsymbol{\theta}_j^T \boldsymbol{x})}{\sum_{k=1}^{K} \exp(\boldsymbol{\theta}_k^T \boldsymbol{x})}$$

Implementing Multi-Class Logistic Regression

- As the model for class K,
  - Use $h_\theta^{(j)}(\boldsymbol{x}) = p(y = j|\boldsymbol{x}; \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_K) = \dfrac{\exp(\boldsymbol{\theta}_j^T \boldsymbol{x})}{\sum_{k=1}^{K} \exp(\boldsymbol{\theta}_k^T \boldsymbol{x})}$
- Gradient descent simultaneously updates all parameters for all models with the above $h_\theta^{(j)}(\boldsymbol{x})$
- Predict class label as the most probable label

$$\max_j h_\theta^{(j)}(\boldsymbol{x}) = \max_j p(y = j|\boldsymbol{x}; \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_K)$$

References

- Andrew Ng, https://www.coursera.org/learn/machine-learning
- http://www.holehouse.org/mlclass/06_Logistic_Regression.html
- Eric Eaton, https://www.seas.upenn.edu/~cis519